Claims

The following is a copy of Applicant's claims that identifies language being added with underlining ("___") and language being deleted with strikethrough ("~~---~~"), as is applicable:

1. (Currently amended) A method for executing a program written for an original computer system on a different host computer system, comprising ~~the steps of~~:

fetching original program ~~code~~ instructions during execution of the program;

dynamically translating the original program ~~code~~ instructions into instructions that can be executed by the host computer system;

dynamically emitting translated program ~~code~~ instructions into at least one code cache of a dynamic execution layer interface of a virtual machine; and

dynamically executing the translated ~~code~~ instructions from ~~within~~ the at least one code cache in lieu of ~~associated~~ the original program ~~code~~ instructions when a semantic function of the ~~associated~~ original program ~~code~~ instructions is requested.

2. (Currently amended) The method of claim 1, wherein ~~the step of~~ fetching program ~~code~~ instructions comprises fetching original program instructions with an emulator.

3. (Original) The method of claim 2, wherein the emulator is an interpreter/emulator.

4. (Currently amended) The method of claim 1, wherein ~~the step of~~ translating the program ~~code~~ instructions comprises translating original program instructions with a just-in-time compiler.

5. (Currently amended) The method of claim 1, wherein ~~the step of~~ emitting translated program ~~code~~ instructions into at least one code cache comprises emitting translated program ~~code~~ instructions into the at least one code cache via an application programming interface.

6. (Currently amended) The method of claim 1, further comprising ~~the step of~~ interpreting and executing program ~~code~~ instructions that ~~has~~ have not be emitted into the at least one code cache.

7. (Currently amended) The method of claim 6, further comprising ~~the step of~~ emulating exception handling actions that would have been performed by the original computer system during execution.

8. (Currently amended) The method of claim 1, further comprising ~~the step of~~, prior to emitting translated program ~~code~~ instructions, growing a code fragment by linking program instructions together.

9. (Currently amended) The method of claim 8, wherein ~~the step of~~ linking program instructions together comprises linking program instructions together with a just-in-time compiler.

10-14. Canceled.

15. (Currently amended) An emulation program configured to emulate an original computer system for which a program was written, the emulation program stored on a computer-readable medium and comprising:

logic configured to dynamically translate original program code into instructions that can be executed by a host computer system;

logic configured to dynamically emit code fragment translations of the original program code into at least one code cache of a dynamic execution layer interface of a virtual machine; and

logic configured to dynamically execute the code fragments within the at least one code cache in lieu of the original program code when a semantic function of the associated program code is requested.

16. (Original) The program of claim 15, wherein the logic configured to translate the program code comprises a just-in-time compiler.

17. (Original) The program of claim 15, wherein the logic configured to emit code fragment translations comprises an application programming interface.

18. (Original) The program of claim 15, further comprising logic configured to interpret and execute program code.

19. (Currently amended) The program of claim 15, further comprising logic configured to emulate exception handling actions of the original computer system for which the program was written.

20. (Currently amended) A system for executing program code that was written for an original computer system on a different host computer system, comprising:

an emulator configured to fetch original program instructions during execution of a program;

a just-in-time ~~translator~~ compiler configured to dynamically translate the original program instructions into instructions that can be executed by the host computer system;

a virtual machine that comprises a dynamic execution layer interface including a core having at least one code cache in which code fragments can be dynamically cached and executed; and

an application programming interface that links the translator to the virtual machine.

21. (Original) The system of claim 20, wherein the emulator comprises an interpreter/emulator.

22. Canceled.

23. (Original) The system of claim 20, wherein the application programming interface comprises a set of functions available to the translator including an emit fragment function with which the translator can emit code fragments into the at least one code cache and an execute function with which the translator can request execution of code fragments contained within the at least one code cache.

24. (Original) The system of claim 23, wherein the application programming interface is configured such that the emit fragment function can be used to perform at least one of tracking a cached code fragment and associating metadata with a cached code fragment.

25. (Original) The system of claim 23, wherein the set of application programming interface functions comprises a lookup fragment function with which cached code fragments can be retrieved.

26. (Original) The system of claim 23, wherein the set of application programming interface functions comprises an invalidate fragment function with which individual cached code fragments can be invalidated.

27. (Original) The system of claim 23, wherein the set of application programming interface functions comprises a enumerate fragment function with which cached code fragments can be enumerated using metadata associated with the fragments.

28. (Original) The system of claim 23, wherein the set of application programming interface functions comprises a cache flush function in which code caches of the dynamic execution layer interface can be flushed.

29. (Original) The system of claim 23, wherein the set of application programming interface functions comprises an install callback function with which the translator can be notified as to events that occur within the dynamic execution layer interface.

30. (Original) An application programming interface configured to link a translator to a dynamic execution layer interface in an computer system emulating system, comprising:

a set of functions available to the translator including:

an emit fragment function with which the translator can emit code fragments into code caches of the dynamic execution layer interface, and

an execute function with which the translator can request execution of code fragments contained within the at least one code cache.

31. (Original) The application programming interface of claim 30, wherein the emit fragment function can be used to perform at least one of tracking a cached code fragment and associating metadata with a cached code fragment.

32. (Original) The application programming interface of claim 30, wherein the set of functions further comprises a lookup fragment function with which cached code fragments can be retrieved.

33. (Original) The application programming interface of claim 30, wherein the set of functions comprises an invalidate fragment function with which individual cached code fragments can be invalidated.

34. (Original) The application programming interface of claim 30, wherein the set of functions comprises a enumerate fragment function with which cached code fragments can be enumerated using metadata associated with the fragments.

35. (Original) The application programming interface of claim 30, wherein the set of functions comprises a cache flush function in which code caches of the dynamic execution layer interface can be flushed.

36. (Original) The application programming interface of claim 30, wherein the set of functions comprises an install callback function with which the translator can be notified as to particular events that occur within the dynamic execution layer interface.

37. (Newly added) The method of claim 1, wherein fetching original program instructions comprises accessing original memory addresses to identify actual locations of the instructions on the host computer system.

38. (Newly added) The method of claim 1, wherein dynamically executing translated instructions comprises dynamically executing translated instructions within the at least one code cache until a cache miss occurs.

39. (Newly added) The method of claim 38, further comprising fetching other original program instructions, dynamically translating the other original program instructions, dynamically emitting further translated program instructions into the at least one code cache, and dynamically executing the further translated program instructions within the at least one code cache.

40. (Newly added) The method of claim 39, further comprising repeating the procedures of claim 39 until substantially all execution of the program comprises execution of translated program instructions within the at least one code cache.